



Fermi National Accelerator Laboratory

TM-1673

On the Electronics for Experiment E687's Trigger on Hadron Momenta

Angelo Cotta Ramusino, Sten Hansen
*Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510*

Dave Buchholz
*Northwestern University
Evanston, Illinois 60201*

July 1990



On the electronics for experiment E687's trigger on hadron momenta.

Angelo Cotta Ramusino, Sten Hansen
FERMILAB Physics Dept. Electronics Support Group
Dave Buchholz
Northwestern University

---- Introduction ----

The purpose of this paper is to describe the electronic modules designed to process the E687 hadron calorimeter's 552 readout channels (ref.1) and generate a trigger signal based upon the total momentum and the total transverse momentum of the detected hadrons.

The trigger system in use for the 1990 data taking period is composed of 4 different types of modules:

- a) Dual 24-inputs mixers (NIM);
- b) Octal gated integrators with input baseline subtraction (NIM);
- c) Final programmable gain mixer (CAMAC);
- d) Dual 8-bit flash ADC and digital discriminator with programmable threshold (CAMAC).

Figure 1 is an artist's view of the system showing schematically how the modules, (except for the digitizers), are interconnected:

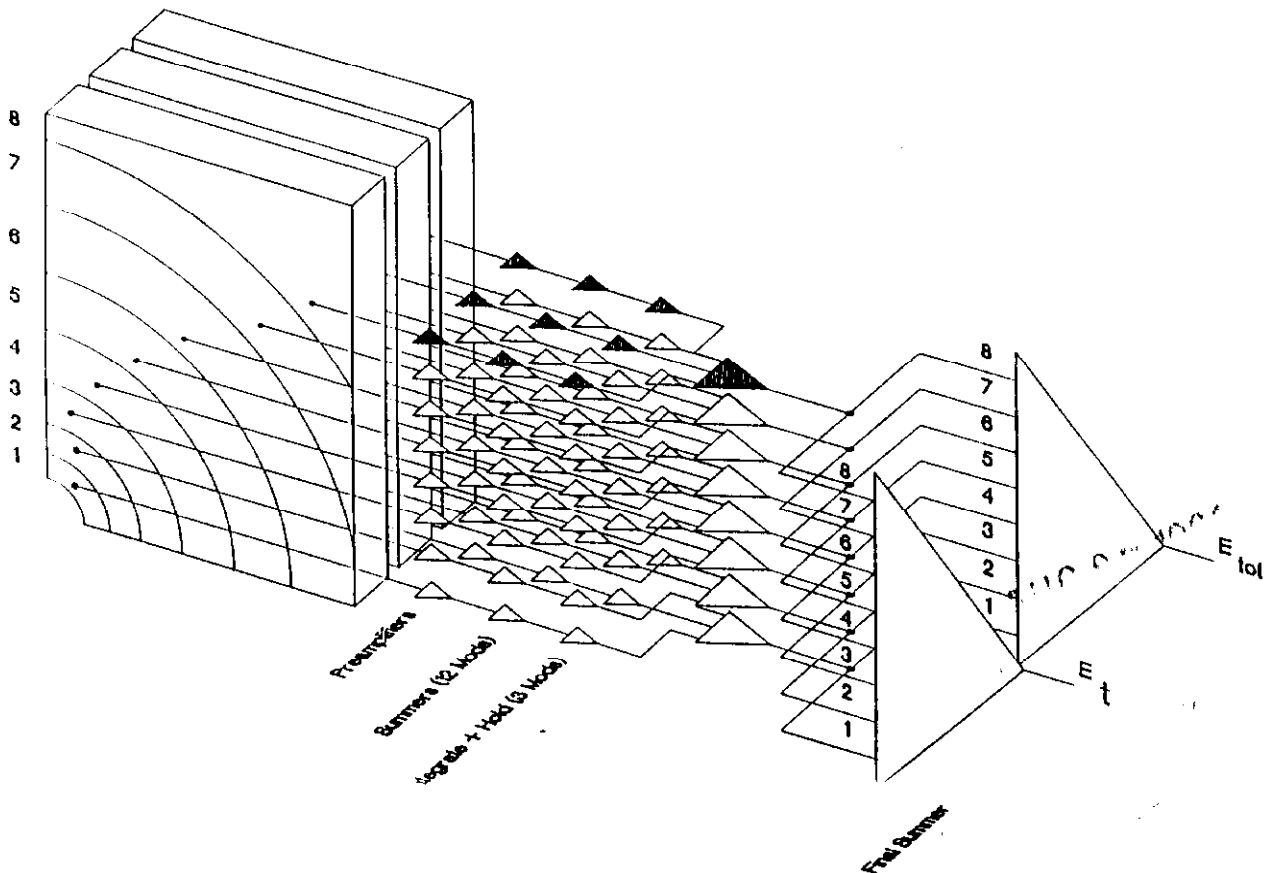


Figure 1

The functions and the characteristics of the modules are briefly outlined in the following paragraphs.

---- Dual 24-inputs mixer ----

This module was designed by the Electronics Support Group in the fall of 1986 for the first run of E687 and it is the only part of the previous trigger system which has not been updated.

The block diagram in Fig.2 shows that the 24 input analog sum is performed by two levels of simpler summing circuits followed by a monolithic output buffer.

A set of gated discriminators is also provided to generate a pulse if any of the analog inputs is above the threshold level set by a front panel mounted potentiometer.

The amplifier used in the simple summing circuit was designed to satisfy the speed and precision requirements of the task. It uses a D.C. servo loop to stabilize the operating point of a discrete wideband preamplifier (Fig.3). The mixer's overall bandwidth is around 30 MHz; the analog propagation delay through the module is around 30nsec; the output voltage swing is +/- 4V into a 50 ohm load; the quiescent power consumption is 7 Watts.

The module is housed in a single width NIM module with two ScotchFlex front panel connectors. The input cables have 26 conductors over a ground plane.

---- Octal gated integrator with input baseline subtraction ----

In the fall of 1988 a new generation of monolithic operational amplifiers with a gain-bandwidth product (GBW) of 50 MHz and upward became available. These amplifiers are widely used in the trigger system modules.

Fig. 4 shows the electronic diagram of the gated integrator module. The digital inputs required are simply a GATE and a RESET (NIM logic) which are converted into TTL logic and are used to operate the integrator switches in the proper sequence. The insert box shows the schematic of a single integrating cell. An input buffer and a difference amplifier stage are added to the usual gated integrator circuit to perform the additional input baseline subtraction function. There is also a general purpose ninth channel, with independent driver logic and input.

The module proves very effective in filtering the E.M.I. noise that couples to the input signals in the experimental hall environment. The circuit tracks the input until the GATE signal is received; the value of the input at the time of the GATE pulse arrival is stored on a capacitor and used as a reference during the integration. Theoretically this reduces the charge error due to a noise waveform of amplitude V_n and frequency f_n to the amount given by:

$$Q_{err} = \{ 3.14 * V_n * f_n * [(T_{gate})^{**2}] \} / R_{int}$$

where: T_{gate} is the integration interval;
 R_{int} is the input resistor of the integrator circuit.

Practically, even a slight mismatch of the resistors in the difference amplifier stage causes a fluctuation of the output BASELINE value as a function of the input baseline.

This degrades the overall performance, but it is within acceptable limits.

As an example, a fluctuation of $\pm 1V$ in the input baseline of a square pulse (1V peak) delivering 58pC to the integrator causes an output error of around 0.5% .

The gate pulse must have a minimum width of 25ns and it must precede the signal by at least 15n. With a 400ns gate the circuit saturates with 20nC input (corresponding to a 3V output voltage) . Please note that the gain and the range of the integrator are determined only by the input termination resistor, 50 Ohm, and by the passive parameters R_{int} and C_{int} ; adapting the module to a different application is just a matter of selecting the proper integrating time constant, $TAU_{int} = R_{int} * C_{int}$, and replacing the components accordingly. The slow op amps in the difference stage cause a propagation delay of about 90ns; although this was tolerable in the present application it could be troublesome in other instances, in which case faster (and more expensive) JFET input op amps should be used. An interval of 150ns must be allowed after a RESET before a new GATE is sent. The module's power consumption is approximately 8 Watt.

---- Final Programmable Mixer ----

The module's schematic diagram in Fig.5 shows that the 24 inputs are subdivided in 8 triplets, corresponding to the hadron energy depositions in the 8 rings of pads in each of the three calorimeter's sections depicted in Fig.1. The first level of three-fold summers mixes the inputs in each triplet, producing 8 signals proportional to the energy deposited in the corresponding cylindrical "rings". The module then sums the 8 "rings" (plus an additional term coming from the Central Hadron Calorimeter, C.H.C.) to produce the total energy signal (E_{tot}).

The Final Programmable Mixer also forms a linear combination of the 8 "rings" with programmable, geometry-dependent weights; if the latter are properly chosen, the result of the linear combination is a signal proportional to the transverse energy (E_{perp}) of the incident hadrons. An additional term in the E_{tot} sum is given by an attenuated version of the signal C.H.C. The attenuation coefficient is also programmable. The MP7524 8-bit CMOS multiplying DAC has been used in the design of the basic programmable attenuator block. The external potentiometer connected between pins 1 and 16 of the chip shunts the internal 10Kohm feedback resistor and the internal parasitic capacitor as well; an increase of the bandwidth is therefore achieved. The gain of the stage becomes dependent on the value of the shunt resistance; in the present version the pot is set so that the maximum transmission of the attenuator is 1/4 (with binary input FF). The gains of the first level mixers were then set to 4 and, as a consequence, the gain of the E_{tot} nine-fold summing stage had to be set to 1/4. An EP900 is used to decode a set of CAMAC commands and a FIFO is used to store the list of programmed attenuation coefficients for readback; appendix A contains a list of the CAMAC functions executed by the module and the EP900 programming equations.

Analog propagation delays thru the different sections of the module have been measured using a -200mV 400ns input pulse with 3ns rise and fall times. The results are listed below:

	prop delay (ns)	rise time (ns)
first level three-fold mixers	7	17
attenuator (input code = FF hex)	18	56
attenuator (input code = 7F hex)	15	50
attenuator (input code = 20 hex)	11	50
output stage (50 Ohm load)	15	21
overall (worst case)	40	70approx

The trimming potentiometers in the circuit allow an accurate channel to channel gain matching to be easily achieved.

The weighting coefficients of the linear combination can be selected in the range 0 to 255/256 in 1/256 steps.

The linear dynamic range of the output is better than +/- 2.5V.

The Final Programmable Mixer's power consumption is 5.5 Watt.

---- Dual flash ADC and programmable trigger logic ----

As shown by the schematic diagram of Fig.6 the analog to digital conversion of the E_{tot} and E_{perp} signals is performed within the module by the two AD9002 8 bit flash ADCs. They are controlled by an ECL sequencer which is started by the leading edge of an ENCODE input pulse. The data from the conversions are compared with programmable threshold values and the comparators' output are processed by a simple logic stage to produce the two outputs E_{tot_OK} and E_{perp_OK} .

A READY flag is provided to validate the E_{tot_OK} and E_{perp_OK} outputs. An ENABLE input provides means of unlocking the module after an acquisition cycle is completed. The digital data and the associated flags are accessible through a front panel differential ECL port.

The sequence of events following the trailing edge of an ENCODE signal is shown by the timing diagram of Fig.7:

- following a module reset the /READY output is unasserted and the E_{tot_OK} and E_{perp_OK} flags are turned off.
- the arriving ENCODE signal is processed to generate a pair of pulses, 5ns wide and 10ns apart, driving the differential CLOCK input of the ADCs; two clock pulses are necessary because the AD9002 works in pipelined mode.
- the ADCs' output data becomes valid 5ns after the second CLOCK pulse, or, typically, 21nsec after ENCODE's leading edge (a total 25nsec response time is assumed for safety).
- the propagation delay through the digital comparators and the following ECL logic is around 10ns and thus the E_{tot_OK} and the E_{perp_OK} outputs become valid 35ns (typ.) from the leading edge of ENCODE.
- READY is asserted after an added 10ns safety margin.

During the regular data taking the module is operated in SINGLE SHOT mode. After one acquisition sequence is complete, an ENABLE pulse is expected before another can be started.

The relationship between an analog input and its digital representation is expressed by the following:

$$\text{eq.1 : } N_{out} = 255 * (|V_{in}| / |V_{ref}|);$$

where: $V_{ref} = 2V.$

The programmable features of the " Dual flash ADC and programmable trigger logic " are detailed in appendix B.

The present version of the module uses two voltage regulators to provide the +/-12V supplies (lacking in the host crate) from the available +/-24V CAMAC power supplies.

---- Conclusions ----

This upgraded trigger system has been installed and operated since the beginning of E687's second run in April 1990. The main advantages over the original version are the 8 bit dynamic range of the flash A/D converters, the improved insensitivity to the E.M.I. noise accompanying the signals and the possibility of setting via software the weights for the determination of E_{perp} (previously accomplished by swapping resistor packs). Since the flash ADC's output is available as a latched data, the experiment can monitor the digitization of the summed hadron calorimeter response.

The trigger decision can be made in as short a time as possible because the total and transverse energy data results from integrating over a 600ns interval whereas the normal integration time for the calorimeter is 800ns.

Fig. 8 shows the trigger efficiency as a function of the programmed threshold for the total hadron energy. The curve is based upon last run's data. Data from the 1990 run, in which the updated trigger system has been employed, is being processed. It is expected that the new efficiency curve will exhibit better " cutoff " characteristics thanks to the improved noise rejection and to the higher resolution of the trigger electronics.

The modules designed for the E687 hadron energy trigger may hopefully have enough general purpose features to render them useful for other applications.

---- Acknowledgements ----

This description of the trigger electronics would not be complete without acknowledging Curtis R. Danner's (Electronic Support Group - Production & Test) excellent job at laying out the gated integrator and the dual adc PC boards (and at bearing the harassment of frequent design modifications).

---- References ----

- 1) S.Park, D.Buchholz, C. Castoldi, D.Claes, B. Gobbi, R. Yoshida ; NorthWestern University, Evanston, Il, U.S.A.
A. Sala ; I.N.F.N.
Sezione di Milano.
V.Arena, G.Boca, A.Cotta Ramusino, R.Diaferia, S.Ratti, P.Vitulo ; Universita' di Pavia - INFN Sezione di Pavia. " Performance of the Hadron Calorimeter for FNAL experiment E687 " NIM A289 (1990) 496-503.

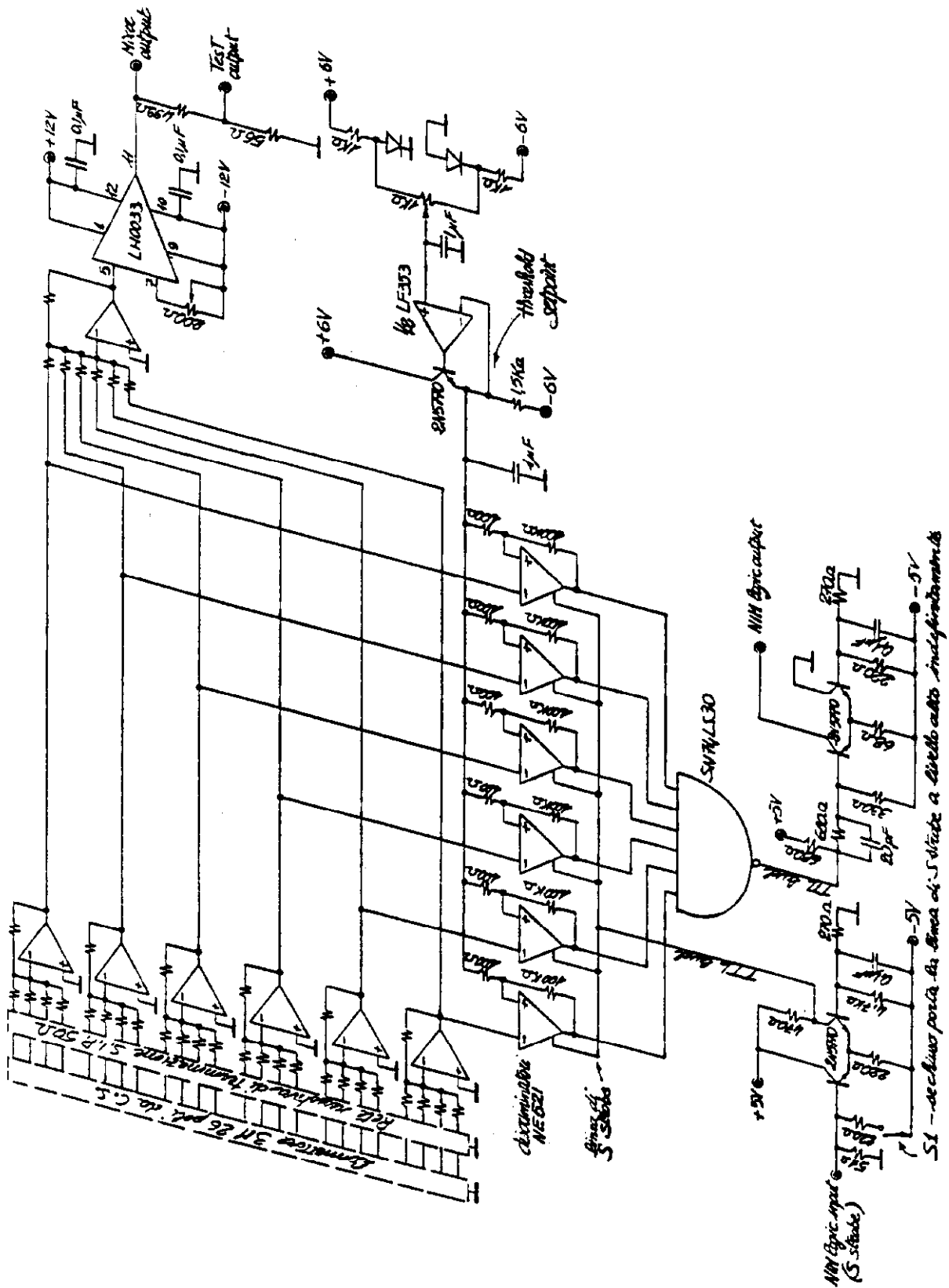
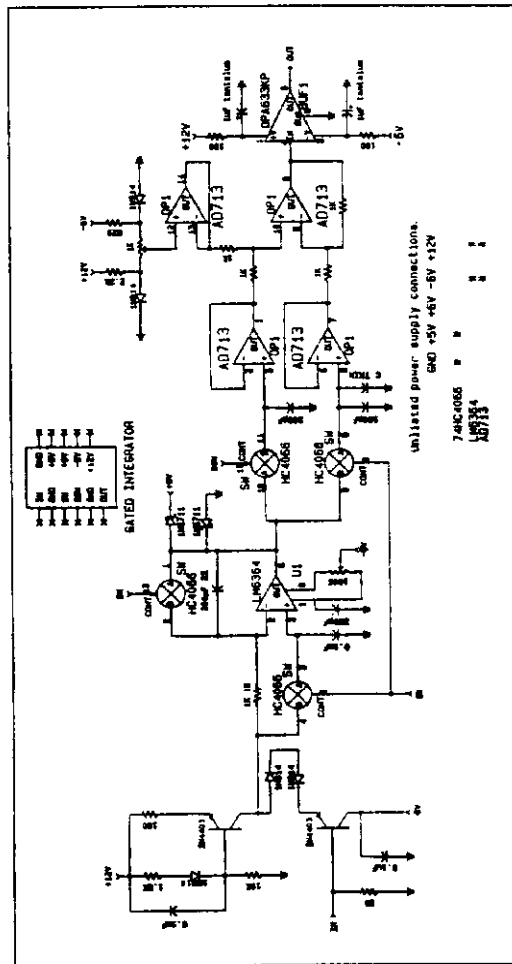
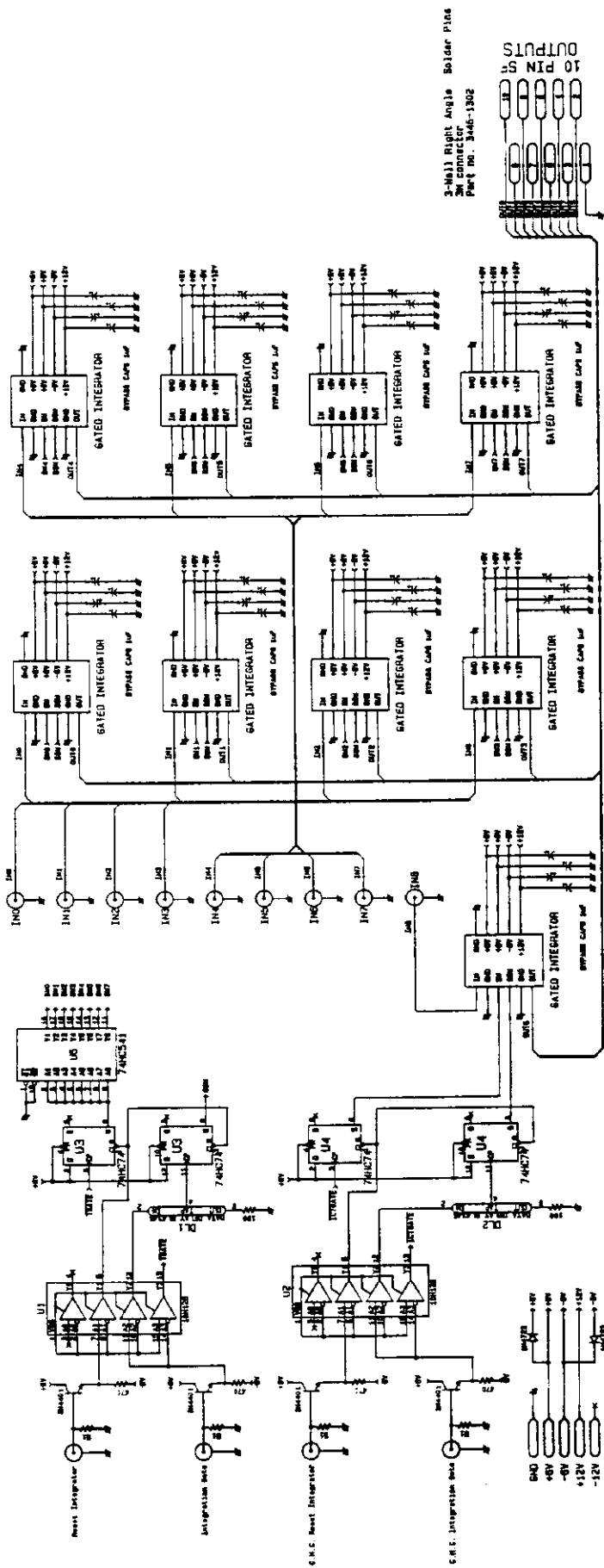


Figure 2




ORIGINATOR	STEN HANSEN, ANGELO COTTA RAMUSINO	SEPT 13 1989
DRAWN	ANGELO COTTA RAMUSINO	JAN 25 1990
REVIEWED	ANGELO COTTA RAMUSINO	
APPROVED		
 FERMI NATIONAL ACCELERATOR LABORATORY UNITED STATES DEPARTMENT OF ENERGY		
E687 HADRON ENERGY TRIGGER UPGRADE: GATED INTEGRATOR WITH INPUT BASELINE SUBTRACTION		
SCALE	DRAWING NUMBER	REV. 3

Figure 4

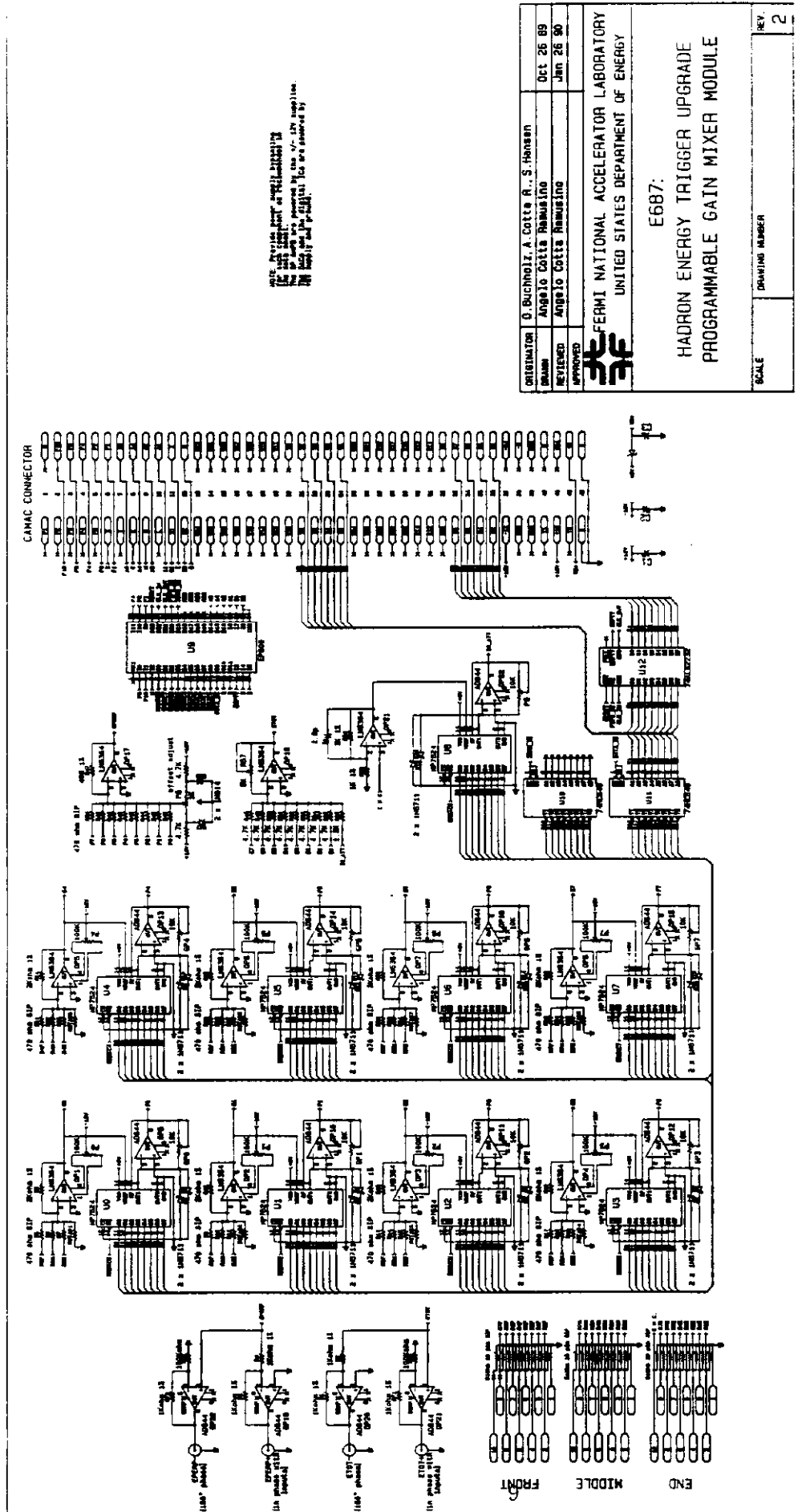
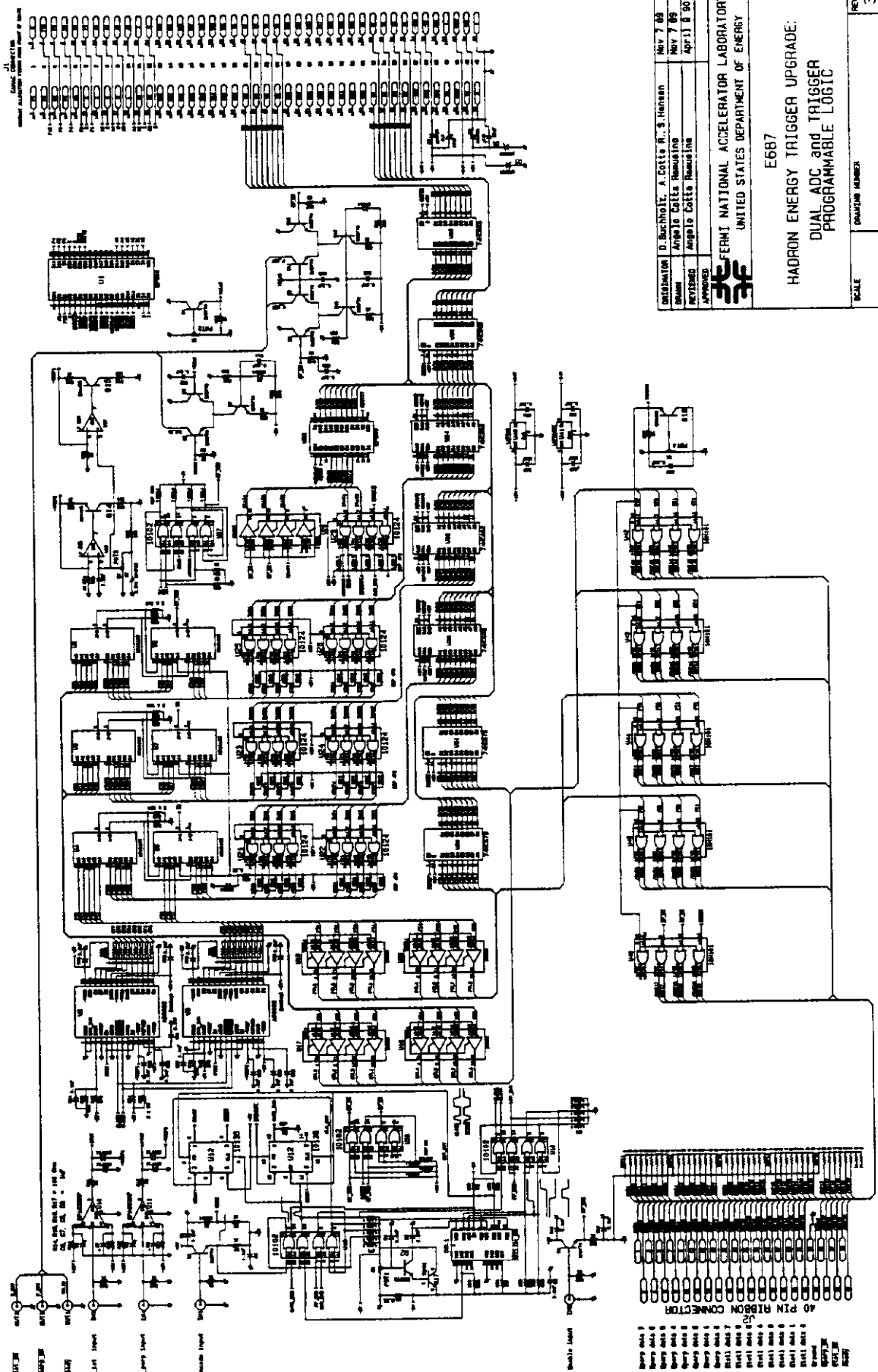


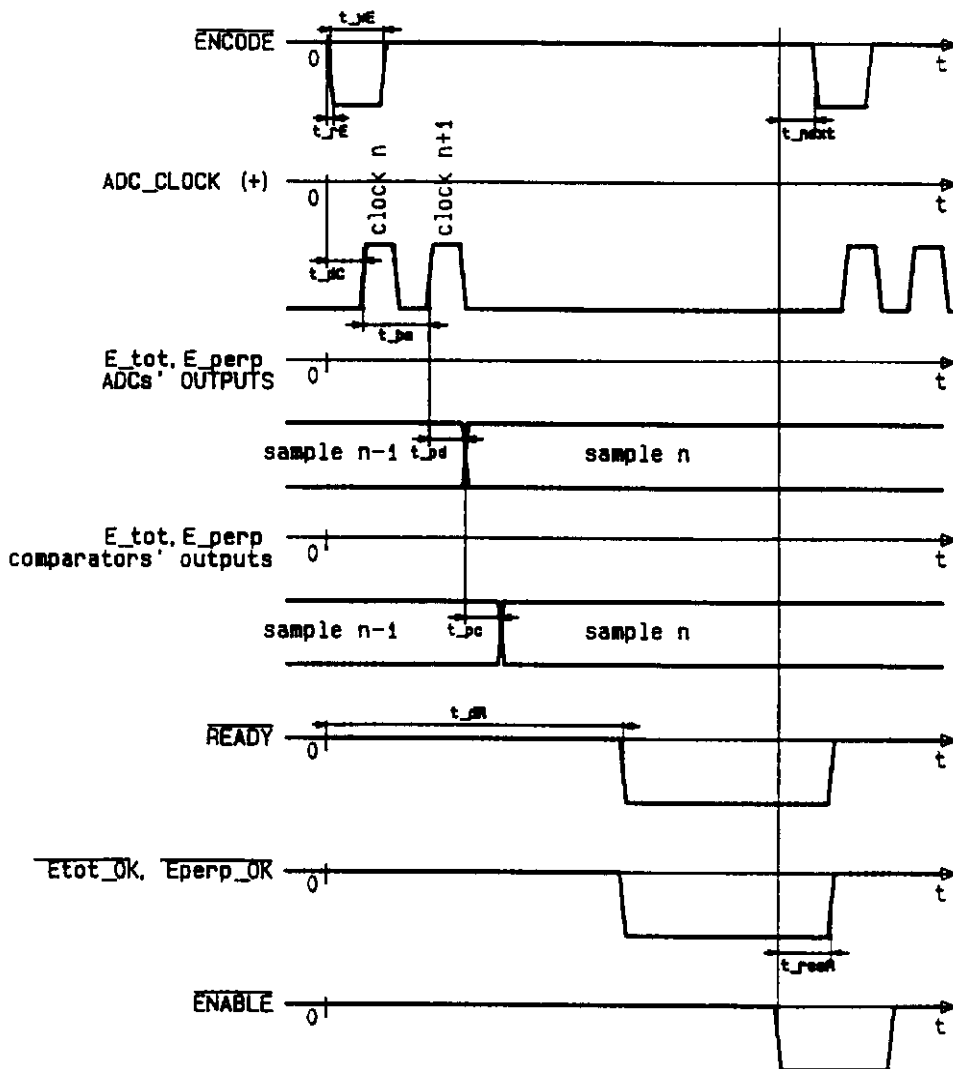
Figure 5



DESIGNATION	D. Buchholz, A. Cotte H. S. Hansen	REV 7 88
REVISION	Angelo Cotte Remaining	REV 7 89
REVISION	Angelo Cotte Remaining	APR 11 8 90
APPROVED		
FERMI NATIONAL ACCELERATOR LABORATORY UNITED STATES DEPARTMENT OF ENERGY		
E687		
HADRON ENERGY TRIGGER UPGRADE: DUAL ADC and TRIGGER PROGRAMMABLE LOGIC		
SCALE	QUANTITY NUMBER	REV 3

Figure 6

Dual ADC and programmable trigger logic module:
conversion timing



Legend:

- t_{re} : ENCODE signal rise time; it must be less than 10ns.
- t_{we} : ENCODE signal pulse width; it must be 10ns or greater.
- t_{next} : time interval following an ENABLE after which an ENCODE pulse can start another conversion = 5ns .
- t_{dc} : delay between the leading edge of ENCODE and the rising edge of the first CLOCK pulse = 6ns.
- t_{bs} : time interval between the CLOCK pulses = 10ns.
- t_{pd} : delay between the second CLOCK pulse and valid data at the output of the ADCs = 5ns.
- t_{pc} : propagation delay through the ECL comparators = 10ns.
- t_{dr} : time interval between the arrival of ENCODE and the assertion of the READY output = 45ns.
- t_{resR} : ENABLE to READY unasserted response time = 8ns.

Figure 7

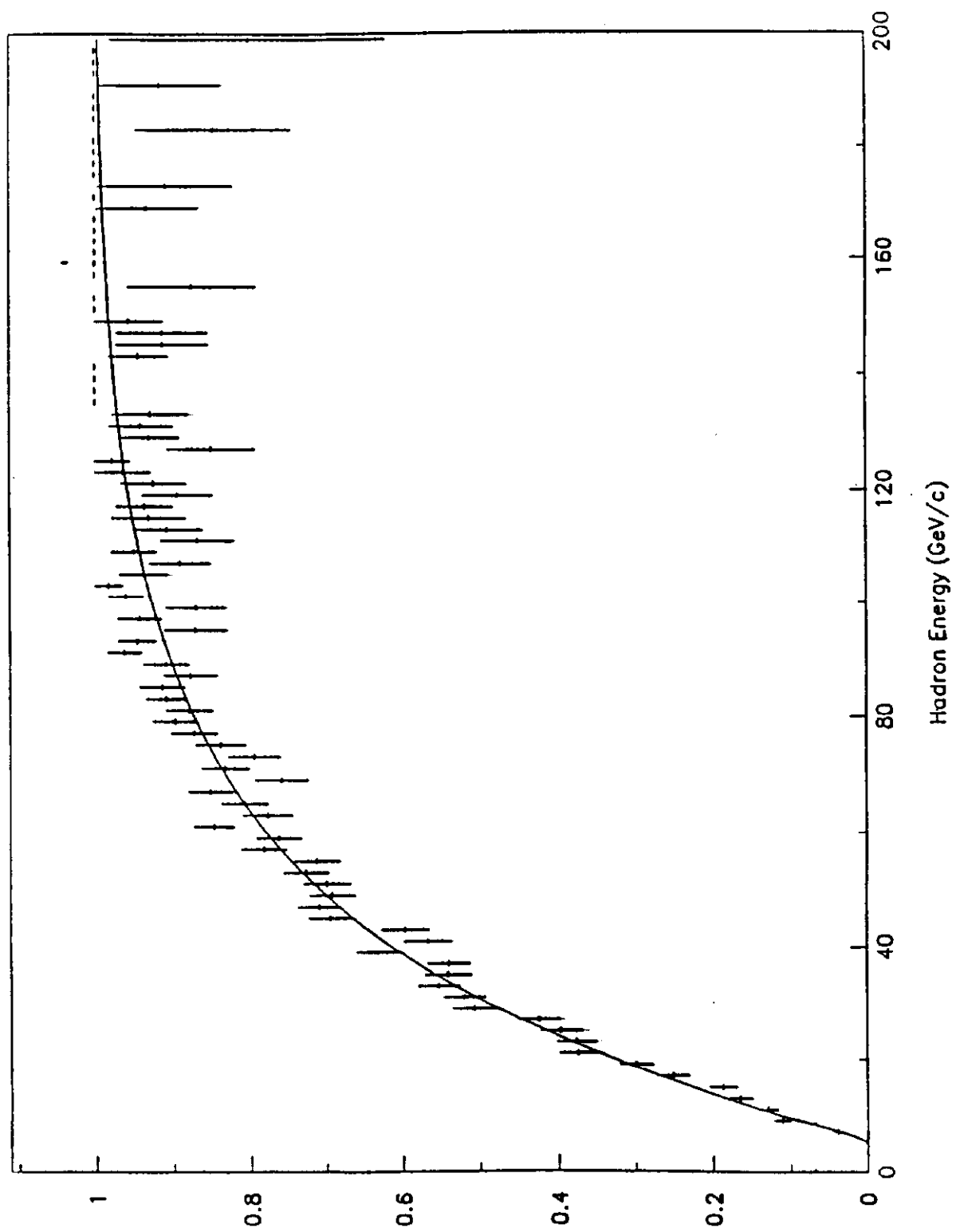


Figure 8

October 26th 1989

Angelo Cotta Ramusino

Fermilab

Physics Dept.

E687 Final Programmable Mixer

The module performs two two-level sums of the 24 inputs, which are driven by the Hadron trigger gated integrators.

The first level sum produces 8 signals, proportional to the energy deposition in the 8 concentric towers in which the detector is subdivided.

The sum of these 8 signals plus a programmable fraction of the C.H.C. input yields the E_tot output. The 8 tower energy signals are also linearly combined in the E_perp output signal, which is proportional to the total transverse momentum of incident hadrons; the relative weight of each tower signal can be programmed in steps of 1/256.

The E_tot and E_perp outputs are available with both positive and negative polarity, to facilitate the interfacing of the programmable gain module with any flash ADC.

The E_tot and E_perp output drivers are capable of +/- 3V voltage swing on a 50 Ohm load.

***** CAMAC functions supported by the module *****

F(16) A(0) : The values of the 9 attenuation coefficients described above may be written to the module in Q-STOP mode block write. The order in which the coefficients are expected is:

- 1) C0 (inner ring)
- 2) C1
- 3) C2
- 4) C3
- 5) C4
- 6) C5
- 7) C6
- 8) C7 (outer ring)
- 9) C.H.C. term attenuation factor.

The module returns Q=TRUE for the first 9 F(16) A(0) function cycles; the module then responds with Q=FALSE to the next F(16) A(0). This tenth access resets an internal pointer/sequencer and makes the module ready for the next block write.

F(0) A(0) : The values of the 9 attenuation coefficients written into the module's latches may be read back with a Q-STOP mode block read operation.

The order in which the 9 words are read out is the same in which they were written.

The module returns Q=TRUE for the first 9 F(0) A(0) commands; the module responds with Q=FALSE to further read commands.

C and Z : The module responds to the two functions in basically the same way:

- 1) the address pointer is reset to the location for coefficient C0.
- 2) the coefficient registers are initialized with FF hex (corresponding to a 255/256 attenuation factor for each attenuator block).

***** Programming equations (ABEL) for the CAMAC decoder EP900 *****

Module Camac_Decoder

Title 'Decodes camac functions for E687 programmable gain mixer module

Angelo Cotta Ramusino Fermilab Physics Department 10/25/89'

p_camac Device 'E0900';

"Inputs:

nN,nF16,nF8,nF4,nF2,nF1	pin	4,3,2,39,38,37;
nA8,nA4,nA2,nA1	pin	27,26,25,24;
nS1,nS2,nS2bis,nZ,nC	pin	23,22,21,19,18;
nEMPTY	pin	17;

"Outputs:

nX,nQ,nZero_OE,nData_OE	pin	16,15,32,14;
P3,P2,P1,P0	pin	31,30,29,28;
FIFOClkIn,FIFOClkOut,FIFO_OE	pin	35,34,33;
nWRDAC0,nWRDAC1,nWRDAC2,nWRDAC3	pin	13,12,11,10;
nWRDAC4,nWRDAC5,nWRDAC6,nWRDAC7	pin	9,8,7,6;
nWRDAC_CHC	pin	5;
nReset	pin	36;

x,z = .X.,.Z.;

Dac_Ptr = [P3,P2,P1,P0];

nWRDAC = [nWRDAC_CHC,nWRDAC7,nWRDAC6,nWRDAC5,
nWRDAC4,nWRDAC3,nWRDAC2,nWRDAC1,nWRDAC0];

F0A0	=	Addr	=	[nF16,nF8,nF4,nF2,nF1,nA8,nA4,nA2,nA1,nN];
F16A0	=	Addr	==	[1,1,1,1,1,1,1,1,1,0];
F16A0	=	Addr	==	[0,1,1,1,1,1,1,1,1,0];

F9	=	Addr	==	[1,0,1,1,0,x,x,x,x,0];
----	---	------	----	------------------------

P3,P2,P1,P0	IsType	'reg_D,feed_reg';
P3.re,P2.re,P1.re,P0.re	IsType	'eqn';
P3.oe,P2.oe,P1.oe,P0.oe	IsType	'eqn';

Equations

" The pointers' clock is the strobe S2; the pointer increments at each write cycle up to address 9 and then it resets to 0; Address 9 is used to generate Q=0 at the end of a block write.

" The pointer is reset by nI directly or by the S2 strobe in a nC cycle.

```

Dac_Ptr    := ( Dac_Ptr + 1 ) & ( Dac_Ptr != 9 ) & F16A0 ;
!P3.re     = nReset ;
!P2.re     = nReset ;
!P1.re     = nReset ;
!P0.re     = nReset ;

P3.oe      = 1;
P2.oe      = 1;
P1.oe      = 1;
P0.oe      = 1;

!nQ        = F16A0 & ( Dac_Ptr != 9 ) # FOA0 & nEMPTY # F9;
nQ.oe      = !nN;

FIFOClkIn  = !nS1 & ( Dac_Ptr != 9 ) & F16A0;

FIFOClkOut = !nS2 & nEMPTY & FOA0;

FIFO_OE    = !nN & FOA0;

!nWRDAC0   = F16A0 & ( Dac_Ptr == 0 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC1   = F16A0 & ( Dac_Ptr == 1 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC2   = F16A0 & ( Dac_Ptr == 2 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC3   = F16A0 & ( Dac_Ptr == 3 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC4   = F16A0 & ( Dac_Ptr == 4 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC5   = F16A0 & ( Dac_Ptr == 5 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC6   = F16A0 & ( Dac_Ptr == 6 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC7   = F16A0 & ( Dac_Ptr == 7 ) & !nS1 # (!nC # !nZ) & !nS2;
!nWRDAC_CHC = F16A0 & ( Dac_Ptr == 8 ) & !nS1 # (!nC # !nZ) & !nS2;

!nX        = ( FOA0 # F16A0 # F9 );
nX.oe      = !nN;

nZero_OE   = nZ & nC;

nData_OE   = ! ( nZ & nC );

nReset     = ( nZ & nC & !F9 ) # nS2;

```

test_vectors

```

( [nF16,nF8,nF4,nF2,nF1,nA8,nA4,nA2,nA1,nN,nS1,nS2,nS2bis,nZ,nC,nEMPTY] ->
  [nX,nQ,Dac_Ptr,FIFOClkIn,FIFOClkOut,FIFO_OE,nWRDAC] )

```

" Initialize cycle :

```

[x,x,x,x,x,x,x,x,x,x,1,1,1,1,0,1,0] -> [z,z,0,0,0,0,0,^B111111111];
[x,x,x,x,x,x,x,x,x,x,1,1,0,0,0,1,0] -> [z,z,0,0,0,0,0,^B000000000];
[x,x,x,x,x,x,x,x,x,x,1,1,1,1,0,1,0] -> [z,z,0,0,0,0,0,^B111111111];

[x,x,x,x,x,x,x,x,x,x,1,1,1,1,1,1,0] -> [z,z,0,0,0,0,0,^B111111111];

```

" Block WRITE :

```

[0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,0,0,0,0,0,0,^B111111111];
[0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,0,0,0,0,0,0,^B111111111];
[0,1,1,1,1,1,1,1,1,0,0,1,1,1,1,0] -> [0,0,0,1,0,0,0,^B111111110];
[0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,0,0,^B111111111];

```



```

[0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,1,9,0,0,0,^B111111111];"60
[0,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1] -> [0,1,9,0,0,0,^B111111111];
[0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,1,9,0,0,0,^B111111111];
[0,1,1,1,1,1,1,1,1,0,1,0,0,1,1,1] -> [0,1,9,0,0,0,^B111111111];
[0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,0,^B111111111];

```

" Block READ :

```

[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,0,0,1,1,1] -> [0,0,0,0,1,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];"70

```

" Let's say the next is the 9th Read of the block

```

[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1] -> [0,0,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,0,0,1,1,1] -> [0,0,0,0,1,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];

[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];"80
[1,1,1,1,1,1,1,1,1,0,1,0,0,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];

```

" Same for further reads:

```

[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,0,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,0,0,1,1,0] -> [0,1,0,0,0,1,^B111111111];
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0] -> [0,1,0,0,0,1,^B111111111];

```

End Camac_Decoder

January 12th 1990

Angelo Cotta Ramusino

Fermilab

Physics Dept.

E687 CAMAC ADC and PROGRAMMABLE TRIGGER LOGIC

The analog to digital conversions of the voltages at the E_{tot} and E_{perp} inputs are performed by the module in response to the leading edge of the ENCODE signal. The two 8-bit words resulting from the simultaneous conversions are compared with three CAMAC programmable threshold values. The comparators' outputs are processed by a simple ECL logic network from which the E_{tot_OK} and E_{perp_OK} signals are derived. E_{tot_OK} and E_{perp_OK} are stable and valid at or after the leading edge of the READY signal and until the arrival of a pulse at the ENABLE input.

A more detailed description of module's features is given below:

- CONVERSION READOUT

The flash ADCs' outputs become valid 25nsec after an ENCODE pulse and remain valid until the beginning of the next conversion cycle. The ADC's outputs can be read by sending to the module the CAMAC commands $F(1)A(0)$ and $F(1)A(1)$ for E_{tot} and E_{perp} respectively. The differential ECL buffered versions of the ADCs' outputs are also available at a front panel connector, for direct readout; the ADC's data is valid at or after the leading edge of the diff.ECL READY flag (pin 39 and 40 of the connector), which can be used to strobe external data latches or to trigger a logic analyzer for diagnostic purposes.

- ONE SHOT or CONTINUOUS CYCLE operating modes

If programmed to operate in ONE SHOT mode the module responds to a new ENCODE input pulse only after being reset by either a front panel ENABLE signal or a CAMAC $F(26)A(X)$ command. In the CONTINUOUS CYCLE mode the module performs a new conversion cycle in response to every ENCODE input pulse, at a maximum rate of 30 MHz.

- CONFIGURABLE DATA PROCESSING

The logic expression for E_{perp_OK} as a function of the E_{tot} and E_{perp} inputs can be either:

a) conditional:

$$E_{perp_OK} = (E_{perp} > P_THRESH) \& (E_{tot} > E_THRESH_P) ;$$

b) unconditional:

$$E_{perp_OK} = (E_{perp} > P_THRESH) ;$$

The logic expression for E_{tot_OK} is instead hard-wired as:

$$E_{tot_OK} = (E_{tot} > E_THRESH) ;$$

NOTE: E_THRESH and E_THRESH_P are DISTINCT values, independently programmable.

- CONTROL/STATUS REGISTER

An 8-bit bidirectional register is provided, which is written

when programming the module's configuration and read when polling the module for the state of internal status flags. The register's bits are mapped according to the following table:

```

bit0  ONE_SHOT   (read/write): ONE_SHOT operating mode selected when HIGH.
bit1  CONDITNL   (read/write): E_perp logic expression a) selected
                                when HIGH.
bit2  READY      (read only)  : Eperp_OK and Etot_OK flags are valid when
                                READY is HIGH.
bit3  Eperp_OK   (read only)  : E_perp trigger signal.
bit4  Etot_OK    (read only)  : E_tot trigger signal.
bit5  -----    : undefined.
bit6  -----    : undefined.
bit7  -----    : undefined.

```

***** CAMAC functions supported by the module *****

```

F(16)A(0) : writes E_THRESH into the dedicated bidirectional latch.
F(16)A(1) : writes P_THRESH into the dedicated bidirectional latch.
F(16)A(2) : writes E_THRESH_P into the dedicated bidirectional latch.
F(16)A(3) : writes the CONTROL/STATUS register.

```

```

F(0)A(0) : reads E_THRESH from the dedicated bidirectional latch.
F(0)A(1) : reads P_THRESH from the dedicated bidirectional latch.
F(0)A(2) : reads E_THRESH_P from the dedicated bidirectional latch.
F(0)A(3) : reads the CONTROL/STATUS register.

```

```

F(1)A(0) : reads E_tot ADC data.
F(1)A(1) : reads E_perp ADC data.

```

F(26)A(X), C, Z : the three functions clear the GATE_INHIBIT flip-flop which prevents retriggering of the conversion cycle when the module is operating in ONE SHOT mode.

NOTE: the front panel NIM signal ENABLE is used to clear the GATE_INHIBIT flip-flop during normal data taking, when the module is regarded as just another component of the D.A. system and read out via the front panel ECL port.

The F(26)A(X) may be used when troubleshooting the module or the system, since then the "dual flash ADC" is preferably read via CAMAC .

***** Programming equations (ABEL) for the CAMAC decoder EP900 *****

Module Camac_Decoder flag '-r3'

Title 'Decodes CAMAC functions for Dual ADC & TRIGGER LOGIC module

Angelo Cotta Ramusino Fermilab Physics Department 02/19/90'

adc_cam Device 'E0900';

"Inputs:

```

nN,nF16,nF8,nF4,nF2,nF1      pin      4,3,2,39,38,37;
nA8,nA4,nA2,nA1      pin      27,26,25,24;
nS1,nS2,nZ,nC      pin      23,22,19,18;

```

"Outputs:

nX,nQ	pin	36,35;
nENABLE	pin	5;
nRESET	pin	16;
nREAD,nWRITE	pin	34,33;
nREADP,nREADE	pin	15,14;
nRDSTAT,WRTSTAT	pin	13,12;
nRDTHE,WRTHE	pin	11,10;
nRDTHPE,WRTHPE	pin	9,8;
nRDTHP,WRTHP	pin	7,6;

x,z = .X.,.Z.;

nX,nQ	IsType	'com,neg';
	Addr	= [nF16,nF8,nF4,nF2,nF1,nA8,nA4,nA2,nA1,nN];
FOA0	= Addr	== [1,1,1,1,1, 1,1,1,1, 0]; " nRDTHE
FOA1	= Addr	== [1,1,1,1,1, 1,1,1,0, 0]; " nRDTHP
FOA2	= Addr	== [1,1,1,1,1, 1,1,0,1, 0]; " nRDTHPE
FOA3	= Addr	== [1,1,1,1,1, 1,1,0,0, 0]; " nRDSTAT
F1A0	= Addr	== [1,1,1,1,0, 1,1,1,1, 0]; " nREADE
F1A1	= Addr	== [1,1,1,1,0, 1,1,1,0, 0]; " nREADP
F16A0	= Addr	== [0,1,1,1,1, 1,1,1,1, 0]; " WRTHE
F16A1	= Addr	== [0,1,1,1,1, 1,1,1,0, 0]; " WRTHP
F16A2	= Addr	== [0,1,1,1,1, 1,1,0,1, 0]; " WRTHPE
F16A3	= Addr	== [0,1,1,1,1, 1,1,0,0, 0]; " WRTSTAT
F9	= Addr	== [1,0,1,1,0, x,x,x,x, 0]; " nRESET
F26	= Addr	== [0,0,1,0,1, x,x,x,x, 0]; " nENABLE

Equations

```
!nX      = ( FOA0  # FOA1  # FOA2  # FOA3
              # F16A0 # F16A1 # F16A2 # F16A3
              # F1A0  # F1A1
              # F26   # F9 );
```

```
nX.oe    = !nN;
```

```
!nQ      = ( FOA0  # FOA1  # FOA2  # FOA3
              # F16A0 # F16A1 # F16A2 # F16A3
              # F1A0  # F1A1
              # F26   # F9 );
```

```
nQ.oe    = !nN;
```

```
nENABLE  = ( nZ & nC & !F26 & !F9 ) # nS2;
```

" nENABLE is used to clear the GATE_INHIBIT and the READY flags before every single-sample acquisition cycle (through F26) or before the start of a continuous acquisition cycle (through F26 or F9). When the board operates in CONTINUOUS CYCLE mode, the nENABLE and the GATE_INHIBIT signals are overlooked.


```

[0,1,1,1,1, 1,1,1,1, 0, 1,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,0,0,1,0];
[0,1,1,1,1, 1,1,1,1, 0, 0,1, 1,1] -> [0,0,1,1,1,1,1,1,1,1,0,0,0,1,0];
[0,1,1,1,1, 1,1,1,0, 0, 1,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,0,0,1,0];
[0,1,1,1,1, 1,1,1,0, 0, 0,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,1,0,0,1,0];
[0,1,1,1,1, 1,1,0,1, 0, 1,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,0,0,1,0];
[0,1,1,1,1, 1,1,0,1, 0, 0,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,1,0,1,0];
[0,1,1,1,1, 1,1,0,0, 0, 1,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,0,0,1,0];
[0,1,1,1,1, 1,1,0,0, 0, 0,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,0,1,1,0];

```

```

[0,0,1,0,1, x,x,x,x, 0, 1,1, 1,1] -> [0,0,1,1,1,1,1,1,1,0,0,0,0,1,1];
[0,0,1,0,1, x,x,x,x, 0, 1,0, 1,1] -> [0,0,0,1,1,1,1,1,1,0,0,0,0,1,1];

```

End Camac_Decoder

module Status_Reg flag '-r3'

Title 'Control/Status Register for Dual ADC & trigger logic module
Angelo Cotta Ramusino Fermilab 01/15/90'
Status device 'E0600';

" Inputs:

```

WRTSTAT, nRDSTAT      pin      1, 14;
READY, nP_OK, nE_OK   pin      6, 7, 8;
nRESET                pin      2;

```

" Outputs:

```

ONESHOT, CONDITNL      pin      4, 5;

```

" Bidirectional:

```

R_W8, R_W7, R_W6, R_W5 pin      15, 16, 17, 18;
R_W4, R_W3, R_W2, R_W1 pin      19, 20, 21, 22;

```

" Set Definitions:

```

DATA = [R_W8,R_W7,R_W6,R_W5,R_W4,R_W3,R_W2,R_W1];

```

H,L,C,X = 1,0,.C,..X.;

```

ONESHOT, CONDITNL      IsType    'reg,feed_pin,pos';
ONESHOT.oe, CONDITNL.oe IsType    'eqn';
R_W8,R_W7,R_W6,R_W5    IsType    'com,feed_pin,neg';
R_W4,R_W3,R_W2,R_W1    IsType    'com,feed_pin,neg';
R_W8.oe,R_W7.oe,R_W6.oe,R_W5.oe IsType    'eqn';
R_W4.oe,R_W3.oe,R_W2.oe,R_W1.oe IsType    'eqn';

```

Equations

```

ONESHOT      := R_W1;
ONESHOT.oe    = 1;
ONESHOT.re    = !nRESET; " nRESET selects continuous cycle.
CONDITNL      := R_W2;
CONDITNL.oe    = 1;
CONDITNL.re    = !nRESET; " nRESET selects unconditional
                        " expression for Eperp.

```

```

R_W8          = 0;
R_W7          = 0;
R_W6          = 0;
R_W5          = !nE_OK;
R_W4          = !nP_OK;
R_W3          = READY;

```

```

R_W2      = CONDITNL;
R_W1      = ONESHOT;

```

```

R_W8.oe = !nRDSTAT;
R_W7.oe = !nRDSTAT;
R_W6.oe = !nRDSTAT;
R_W5.oe = !nRDSTAT;
R_W4.oe = !nRDSTAT;
R_W3.oe = !nRDSTAT;
R_W2.oe = !nRDSTAT;
R_W1.oe = !nRDSTAT;

```

test_vectors

```

([WRTSTAT,nRDSTAT,READY,nP_OK,nE_OK,nRESET,
 !R_W8,!R_W7,!R_W6,!R_W5,!R_W4,!R_W3,!R_W2,!R_W1]
-> [ONESHOT,CONDITNL,!R_W8,!R_W7,!R_W6,!R_W5,!R_W4,!R_W3,!R_W2,!R_W1])

```

```

[0,1,0,1,1,1,X,X,X,X,X,X,X,X] -> [X,X,X,X,X,X,X,X,X,X,X,X];"
[1,1,0,1,1,1,X,X,X,X,X,X,1,1] -> [0,0,X,X,X,X,X,X,X,X,X,X];
[0,1,0,1,1,1,X,X,X,X,X,X,X,X] -> [X,X,X,X,X,X,X,X,X,X,X,X];
[1,1,0,1,1,1,X,X,X,X,X,X,1,0] -> [1,0,X,X,X,X,X,X,X,X,X,X];
[0,1,0,1,1,1,X,X,X,X,X,X,X,X] -> [X,X,X,X,X,X,X,X,X,X,X,X];"
[1,1,0,1,1,1,X,X,X,X,X,X,0,1] -> [0,1,X,X,X,X,X,X,X,X,X,X];
[0,1,0,1,1,0,X,X,X,X,X,X,X,X] -> [0,0,X,X,X,X,X,X,X,X,X,X];
[0,1,0,1,1,1,X,X,X,X,X,X,X,X] -> [0,0,X,X,X,X,X,X,X,X,X,X];
[1,1,0,1,1,1,X,X,X,X,X,X,0,0] -> [1,1,X,X,X,X,X,X,X,X,X,X];

```

```

[0,0,0,1,1,1,X,X,X,X,X,X,X,X] -> [1,1,X,X,X,1,1,1,0,0,0];"
[C,1,1,1,1,1,X,X,X,X,X,X,1,1] -> [0,0,X,X,X,X,X,X,X,X,X,X];
[C,1,0,1,1,1,X,X,X,X,X,X,0,0] -> [1,1,X,X,X,X,X,X,X,X,X,X];
[0,0,1,1,1,1,X,X,X,X,X,X,X,X] -> [1,1,X,X,X,1,1,0,0,0,0];
[0,0,0,1,0,1,X,X,X,X,X,X,X,X] -> [1,1,X,X,X,0,1,1,0,0,0];

```

End Status_Reg